

fission

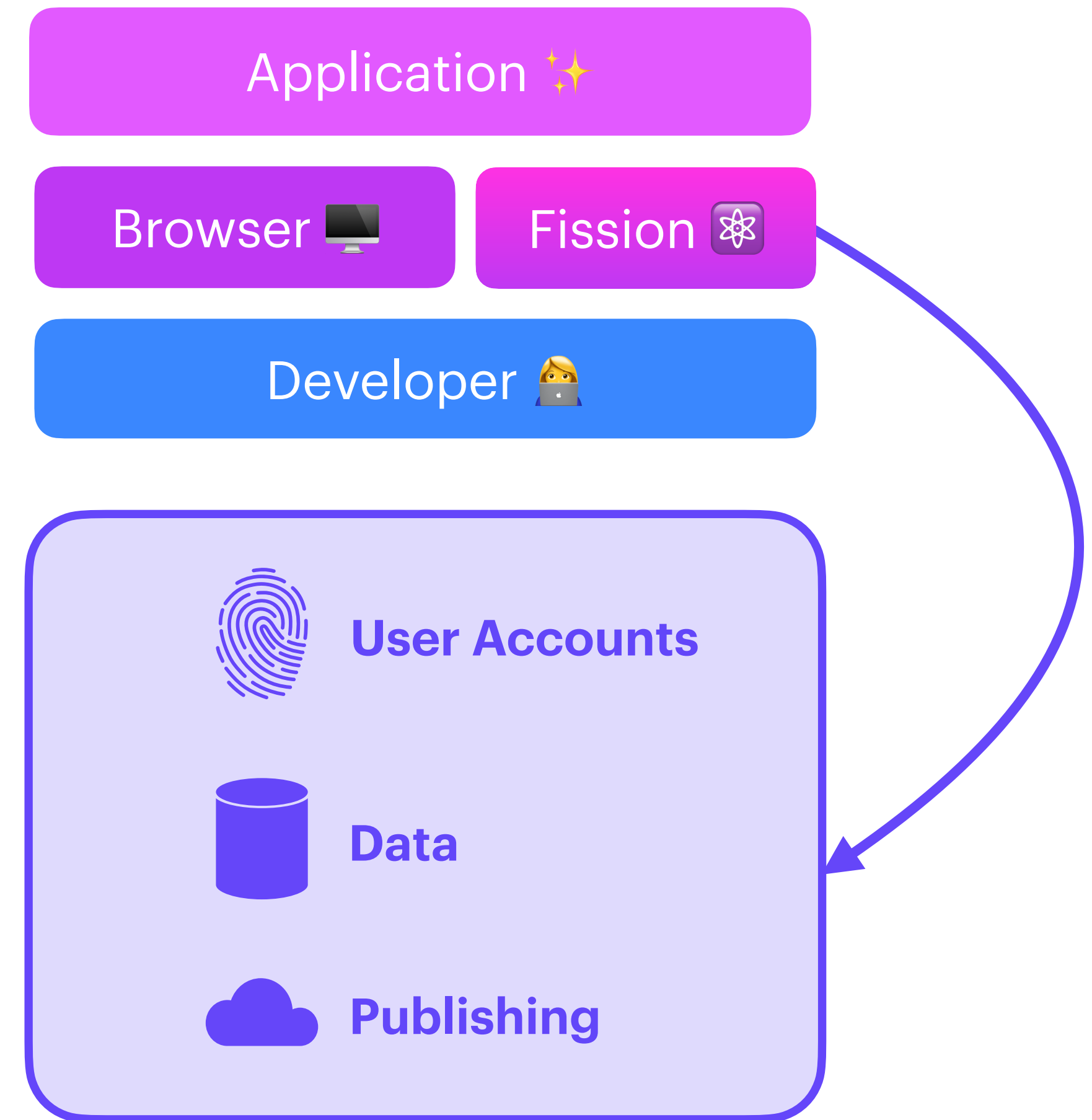
FISSION FOR ENGINEERING TEAMS

May 2021



WHAT IS FISSION?

- Fission is building a client side edge stack with identity & auth, user owned data, and a distributed file system on top of IPFS
- Standardize and integrate "hard" things like privacy, security, sync & more
- Individuals and teams can focus on full featured apps that act like desktop or mobile, including offline





OVERVIEW

 Technology building blocks

 The Fission Whitepaper

 Webnative SDK

 The App Publishing Platform



TECH BUILDING BLOCKS

- **IPFS:** Content-addressed, peer-to-peer storage network
- **Browser APIs:** WebCrypto, Service Workers, IndexedDB and PWAs
- **Decentralized Identifiers (DIDs):** portable accounts that can be user owned, plus Verifiable Credentials for data linked to accounts



THE FISSION WHITEPAPER

Authentication

DID (key method) self-certified.

Authorization

Query access : Reference (CID) + Decryption Key

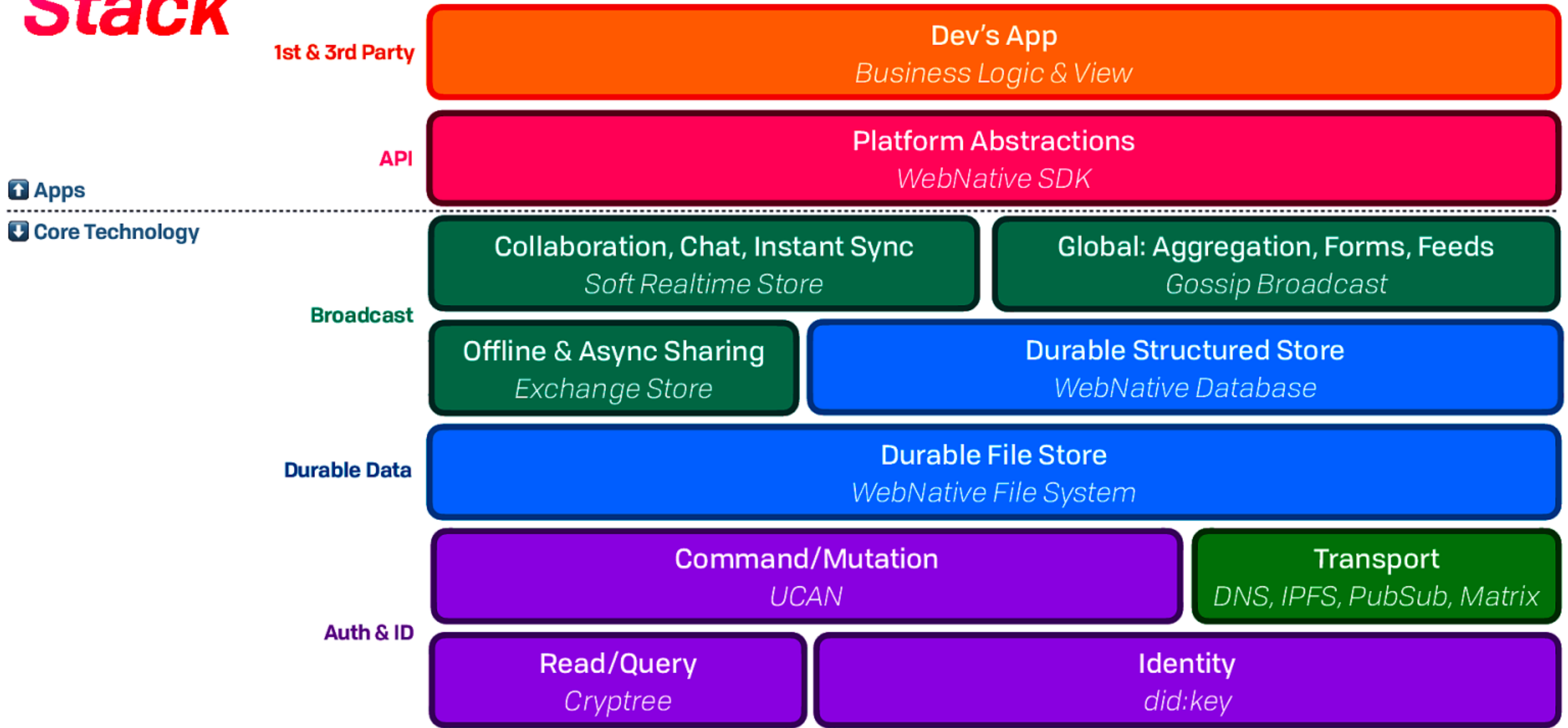
Command + Mutation: **UCAN** - self-contained tokens (JWT)

Webnative Filesystem

WNFS versioned, logged, programmable, secure and user-controlled

Full details: <https://whitepaper.fission.codes/>

WebNative 
Stack





WEBNATIVE SDK

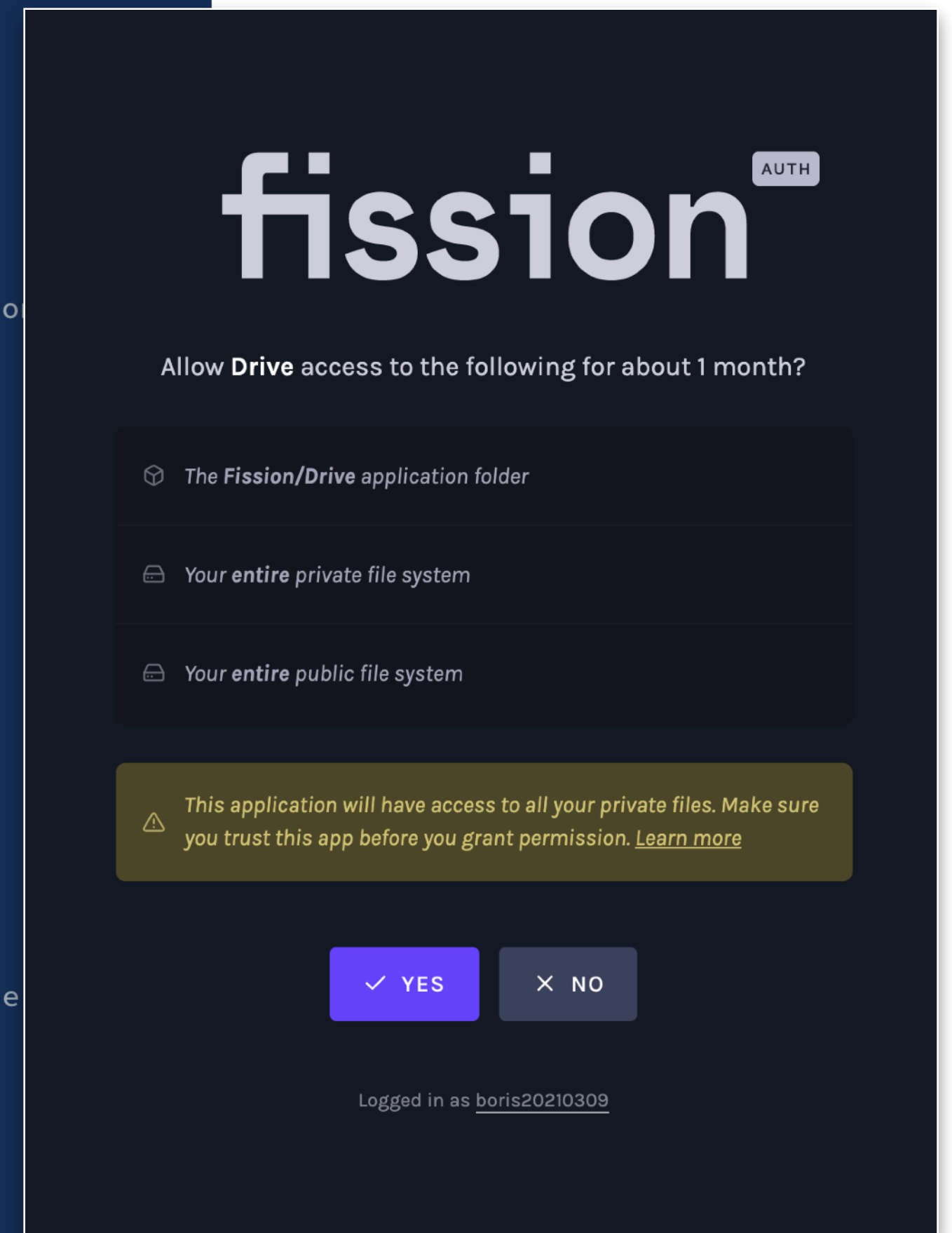
- Decentralized, offline, user-controlled web app development
- Enables password-less, secure authentication
- Provides encrypted-at-rest file storage, powered by IPFS.
- Documentation: <https://guide.fission.codes/developers/webnative>
- Open Source ([@fission-suite/webnative](https://github.com/fission-suite/webnative))



🔑 AUTHENTICATION + AUTHORIZATION

- Passwordless authentication
- Private Keys generated / managed by the WebCrypto API
- Authorization granted via UCAN

```
1  const state = await wn.initialise({
2    permissions: {
3      app: {
4        name: "Demo",
5        creator: "Fission"
6      },
7
8      // Ask the user permission for additional
9      fs: {
10       privatePaths: [ "Music" ],
11       publicPaths: [ "Mixtapes" ]
12     }
13   }
14 })
15
16
17  switch (state.scenario) {
18    case wn.Scenario.AuthCancelled:
19      // User was redirected to lobby,
20      // but cancelled the authorisation
21      break;
22
23    case wn.Scenario.AuthSucceeded:
24    case wn.Scenario.Continuation:
25      // We can now interact with our file
26      state.fs
27      break;
28
29    case wn.Scenario.NotAuthorised:
30      wn.redirectToLobby(state.permissions)
31      break;
32
33  }
```





- IPFS powered filesystem
 - Public Files: available across the IPFS network
 - Private Files: encrypted-at-rest on IPFS
- Familiar UNIX-style API
- Full version history

```
1 // After initialising ...
2 const fs = state.fs
3 const appPath = fs.appPath()
4
5 // List the user's private files that belong to this
6 await fs.ls(appPath)
7
8 // Create a sub directory
9 await fs.mkdir(fs.appPath([ "Sub Directory" ]))
10 await fs.publish()
```



APP PUBLISHING PLATFORM

- Static file (JAMStack) publishing
- Powered by IPFS: Peer-to-Peer distributed filesystem
- Free subdomain + SSL
- Custom domains
- MicroSaaS: apps can programmatically build new apps



CLI

- Full app publishing from any directory
- Bundled go-ipfs (no additional install required)
- Documentation: <https://guide.fission.codes/developers/cli>
- Open Source ([@fission-suite/fission](https://github.com/fission-suite/fission))



CLI: AUTHENTICATION

- **fission setup**
 - Downloads and starts local IPFS daemon.
 - Registers an account (generating a private key)
 - Creates DNSLink for user's DID

```
1 $ fission setup
2 🌱 Setting up environment
3 🍌 Downloading managed IPFS for Linux
4 🗄️ Configuring managed IPFS
5 generating ED25519 keypair...done
6 peer identity: 12D3KooWHZ3C1tDmzUhgi3A6LYmgBxvv1E1obtG7S2ZkYM
7 initializing IPFS node at /home/yourname/.config/fission/ipfs
8 🗝️ Creating keys
9 🏠 Do you have an existing account? [Y/n] n
10 Username: YOURNAME
11 Email: yourname@example.com
12 ✅ Registration successful! Head over to your email to confirm
13 🗄️ Initializing user config file
14 ✅ Done! Welcome to Fission, YOURNAME ✨
```



CLI: APP PUBLISHING

- **fission app register**
 - Set custom build directory
 - Generate subdomain (with SSL)
 - Creates local fission.yaml config file
- **fission app publish**
 - Add files to IPFS (replicated to Fission servers), update DNSLink

```
1 $ fission app register
2 🧑 Build directory (./dist):
3 ✅ App initialized as big-narrow-fuchsia-elf.fission.app
4 🎬 Next run fission app publish or fission app publish
5 🧑 It may take DNS time to propagate this initial set
6 you can always view your app at
7 https://ipfs.runfission.com/ipns/big-narrow-fuchsia-elf.fission.app
```

```
1 $ fission app publish
2 🚀 Now live on the network
3 📄 DNS updated! Check out your site at:
4 🔗 big-narrow-fuchsia-elf.fission.app
```



GITHUB ACTION

- Built on top of CLI
- Enables multi-user publishing to a single app
- Trigger deployments (e.g. Headless UI)
- Open Source ([@fission-suite/publish-action](https://github.com/fission-suite/publish-action))

Publish
succeeded 7 days ago in 1m 38s

- > ✓ Set up job
- > ✓ Build fission-suite/publish-action@v1
- > ✓ Checkout repository
- > ✓ Run actions/setup-node@v2
- > ✓ Get cache directory
- > ✓ Cache modules
- > ✓ Install packages
- > ✓ Build assets
- > ✓ Publish to Fission!
- > ✓ Post Cache modules
- > ✓ Post Checkout repository
- > ✓ Complete job



WEBNATIVE: APP PUBLISHING

- Create new apps programatically
- Publish content and updates from a Web-based UI
- All the functionality of the CLI in the browser.

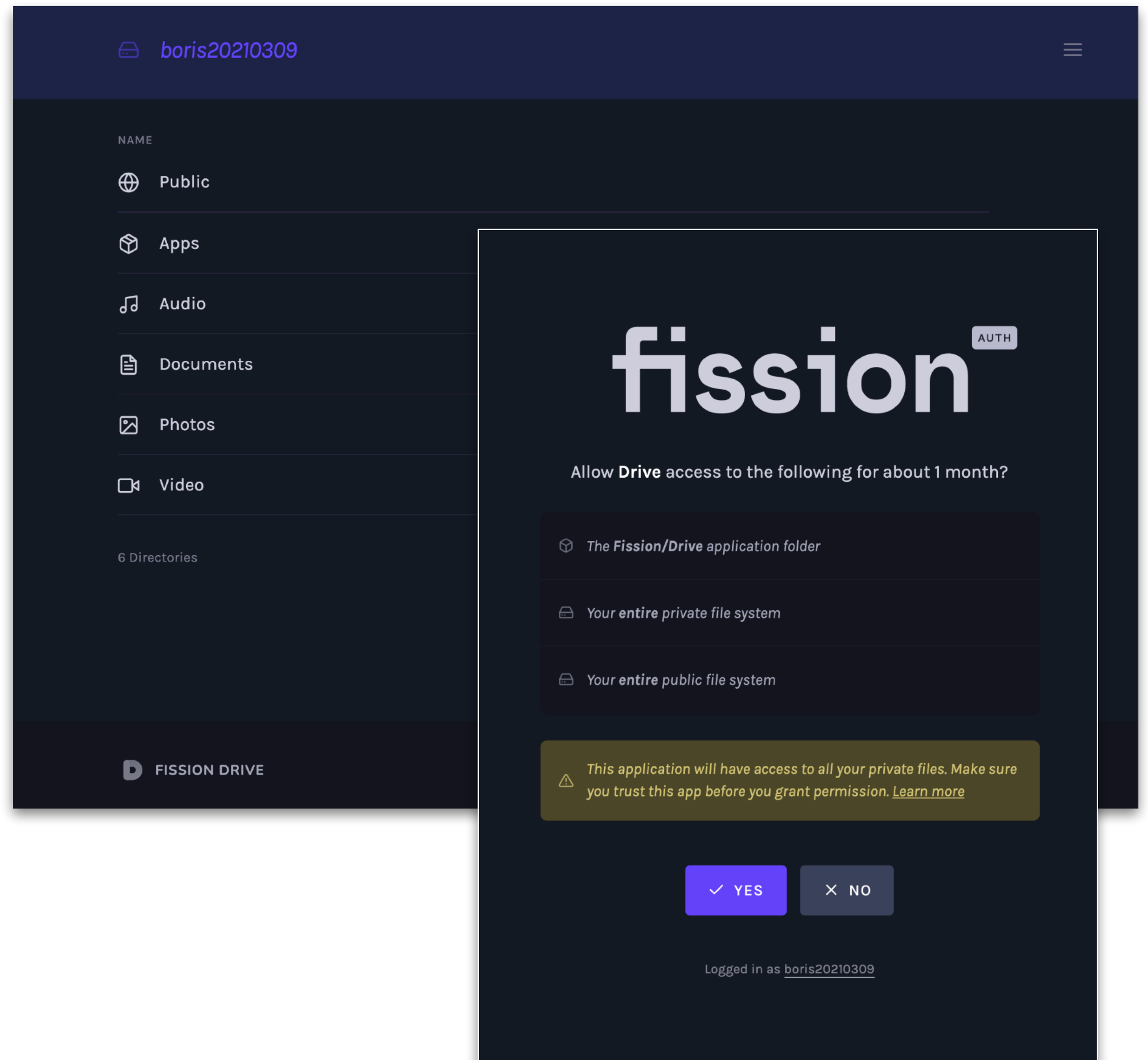
```
1 const newApp = await sdk.apps.create()  
2 // 'your-fission-deployment.fission.app'
```

```
await sdk.apps.publish('your-fission-deployment.fission.ap  
| 'QmRVvvMeMEPi1zerpXYH9df3ATdzuB63R1wf3Mz5NS5HQN')
```



FISSION DRIVE

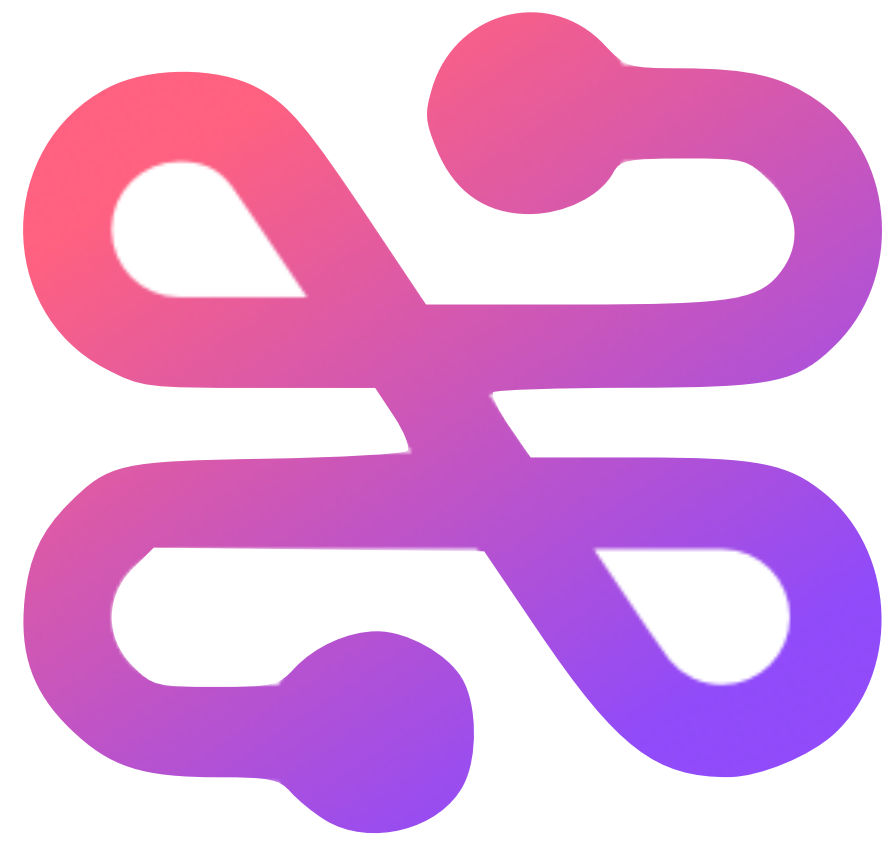
- Default WNFS file browser
- Users can view App Data
- Encrypted by default, public folder is available unencrypted with direct IPFS links
- Works offline
- <https://drive.fission.codes>





HOW TO WORK WITH FISSION

- **Live today!**
 - Go to <https://guide.fission.codes>, install the CLI and start building an app
 - Visit <https://drive.fission.codes> to create a Fission account and try out Fission Drive
- **Book an engineering briefing for your team:** we want you to succeed building with Fission, our Developer Success team is here to support you in shipping apps
- **Feature acceleration:** see a feature on the roadmap you'd like to have access to sooner? Have a particular integration in mind? Get in touch about scoping a project.
- **Run a Fission Constellation:** talk to us about running our app publishing platform as a turn key developer experience for your protocol or community



fission

THANK YOU!

Github: [fission-suite](https://github.com/fission-suite)

Developer Guide & Docs:
guide.fission.codes

Technical Whitepaper:
whitepaper.fission.codes 🚧 🧑🏻‍🔧 ✍️

Home page: fission.codes

Discourse forum: talk.fission.codes

Discord chat: fission.codes/discord

Twitter: [@fissioncodes](https://twitter.com/fissioncodes)



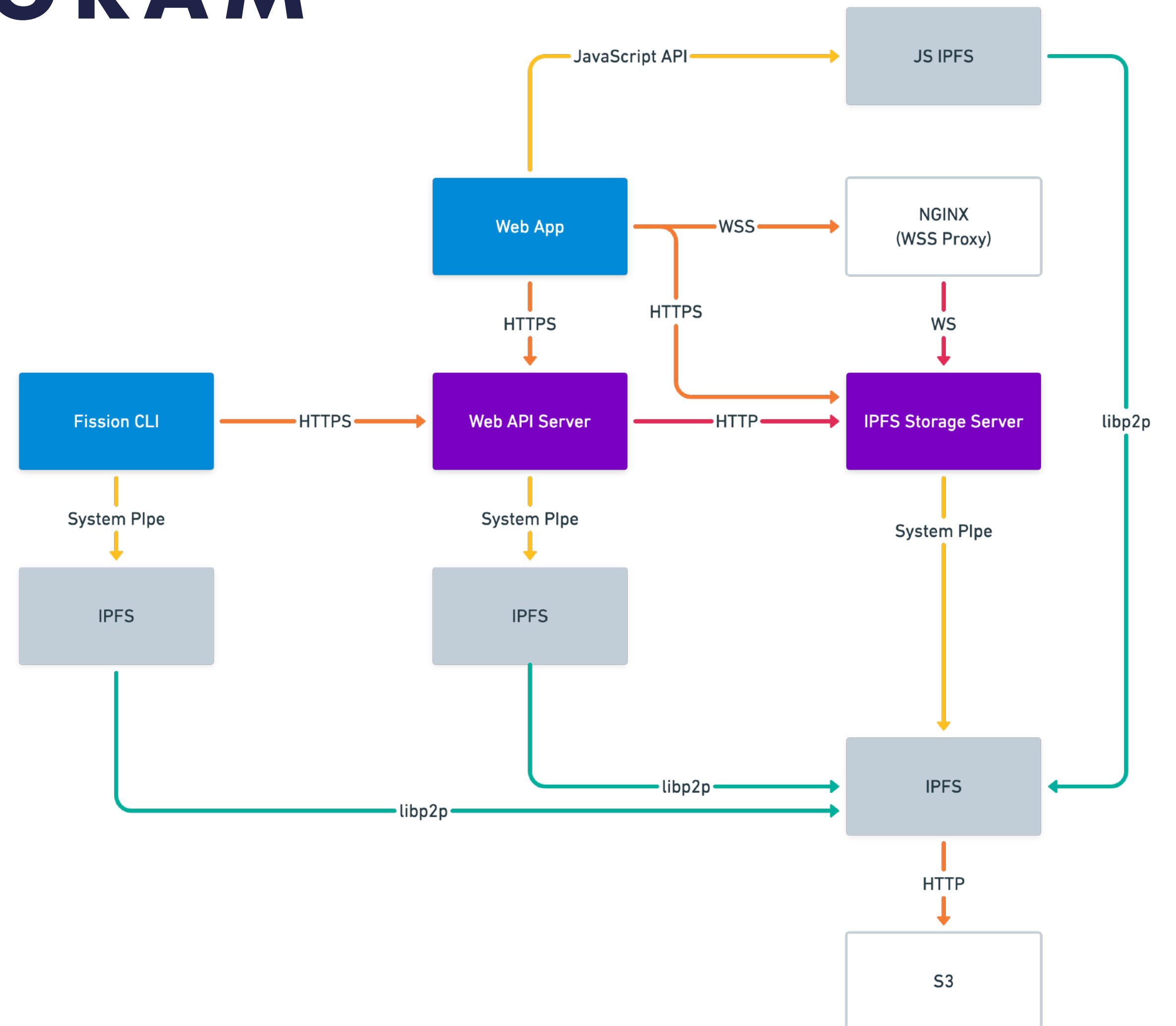
CONSTELLATION: APP PUBLISHING PLATFORM IN A BOX

- Our app publishing platform is open source and can be federated (a constellation provider rather than a cloud provider)
- We run a Fission instance with *.fission.app and *.fission.name and integrate everything for many smaller developers; focused on mass market front end developers
- Communities or Organizations can run their own Fission Constellation, choose and customize their own domain names and features



FISSION SYSTEM DIAGRAM

- Fission Web API (Haskell)
- Auth Lobby Web App (Elm)
- Fission CLI (Haskell)
- Drive Web App (Elm)
- Integrations, eg. Publish action on Github, Heroku Add-On, etc.



PRIVATE KEYS IN THE BROWSER

- As part of an initial grant from Protocol Labs and Filecoin, we have developed an extension to webnative that allows for secure, non-custodial storage of private keys in browser, without plugins, including on mobile
- It uses BLS for signature aggregation and a co-signer model
- Keys can be generated on the fly in browser for new users, with instant onboarding
- Available now for Filecoin keys, Ethereum keys are next on the roadmap